Table of Contents

1. Introduction
2. GitHub
    i. Ignore Whitespace
    ii. Adjust Tab Space
    iii. Commit History by Author
    iv. Cloning a Repository
    v. Compare all Branches to Another Branch
    vi. Comparing Branches
    vii. Compare Branches across Forked Repositories
    viii. Gists
    ix. Git.io
    x. Keyboard Shortcuts
    xi. Line Highlighting in Repositories
    xii. Closing Issues via Commit Messages
    xiii. Cross-Link Issues
    xiv. CI Status on Pull Requests
    xv. Syntax Highlighting in Markdown Files
    xvi. Emojis
    xvii. Images/GIFs
        i. Embedding Images in GitHub Wiki
    xviii. Quick Quoting
    xix. Quick Licensing
    xx. Task Lists
        i. Task Lists in Markdown Documents
    xxi. Relative Links
    xxii. Metadata and Plugin Support for GitHub Pages
    xxiii. Viewing YAML Metadata in your Documents
    xxiv. Rendering Tabular Data
    xxv. Diffs
        i. Rendered prose Diffs
        ii. Diffable Maps
        iii. Expanding Context in Diffs
        iv. Diff or Patch of Pull Request
        v. Rendering and diffing images
    xxvi. Hub
    xxvii. Decreasing Contributor Friction
    xxviii. Contributing Guidelines
    xxix. GitHub Resources
        i. GitHub Talks
3. Git
    i. Previous Branch
    ii. Stripspace
    iii. Checking out Pull Requests
    iv. Empty Commits :trollface:
    v. Styled Git Status
    vi. Styled Git Log
    vii. Git Query
    viii. Merged Branches
    ix. Web Server for Browsing Local Repositories
    x. Git Configurations
        i. Aliases
        ii. Auto-Correct
        iii. Color

xi. Git Resources
    i. Git Books

GitHub Cheat Sheet

A collection of cool hidden and not so hidden features of Git and GitHub. This cheat sheet was inspired by Zach Holman's Git and GitHub Secrets talk at Aloha Ruby Conference 2012 (slides) and his More Git and GitHub Secrets talk at WDCNZ 2013 (slides).

Shortlink: `http://git.io/sheet`

Read this in other languages: English, 한국어, 日本語, 简体中文.

Table of Contents

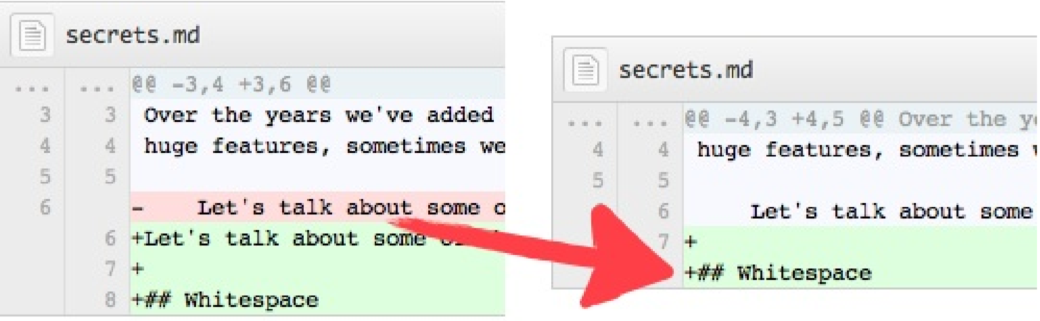
- GitHub
  - Ignore Whitespace
  - Adjust Tab Space
  - Commit History by Author
  - Cloning a Repository
  - Compare all Branches to Another Branch
  - Comparing Branches
  - Compare Branches across Forked Repositories
  - Gists
  - Git.io
  - Keyboard Shortcuts
  - Line Highlighting in Repositories
  - Closing Issues via Commit Messages
  - Cross-Link Issues
  - CI Status on Pull Requests
  - Syntax Highlighting in Markdown Files
  - Emojis
  - Images/GIFs
    - Embedding Images in GitHub Wiki
  - Quick Quoting
  - Quick Licensing
  - Task Lists
    - Task Lists in Markdown Documents
  - Relative Links
  - Metadata and Plugin Support for GitHub Pages
  - Viewing YAML Metadata in your Documents
  - Rendering Tabular Data
  - Diffs
    - Rendered prose Diffs
    - Diffable Maps
    - Expanding Context in Diffs
    - Diff or Patch of Pull Request
    - Rendering and diffing images
  - Hub
  - Decreasing Contributor Friction
  - Contributing Guidelines
  - GitHub Resources
    - GitHub Talks
- Git
  - Previous Branch
  - Stripspace
  - Checking out Pull Requests
  - Empty Commits :trollface:

- Styled Git Status
- Styled Git Log
- Git Query
- Merged Branches
- Web Server for Browsing Local Repositories
- Git Configurations
  - Aliases
  - Auto-Correct
  - Color
- Git Resources
  - Git Books

GitHub

Ignore Whitespace

Adding `?w=1` to any diff URL will remove any changes only in whitespace, enabling you to see only that code that has changed.

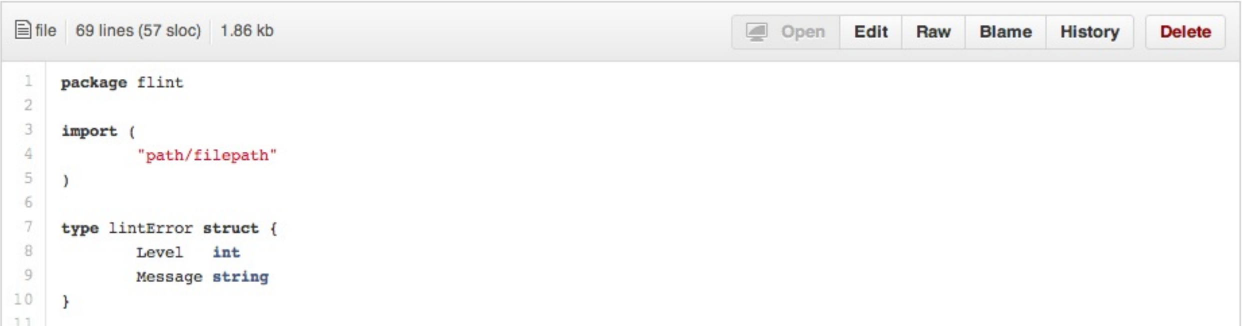


Read more about GitHub secrets.

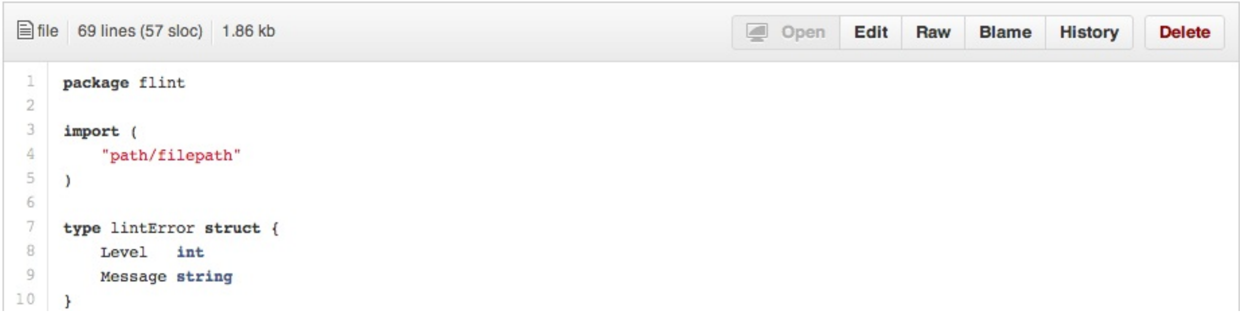
Adjust Tab Space

Adding `?ts=4` to a diff or file URL will display tab characters as 4 spaces wide instead of the default 8. The number after `ts` can be adjusted to suit your preference. This does not work on Gists, or raw file views.

Here is a Go source file before adding `?ts=4`:



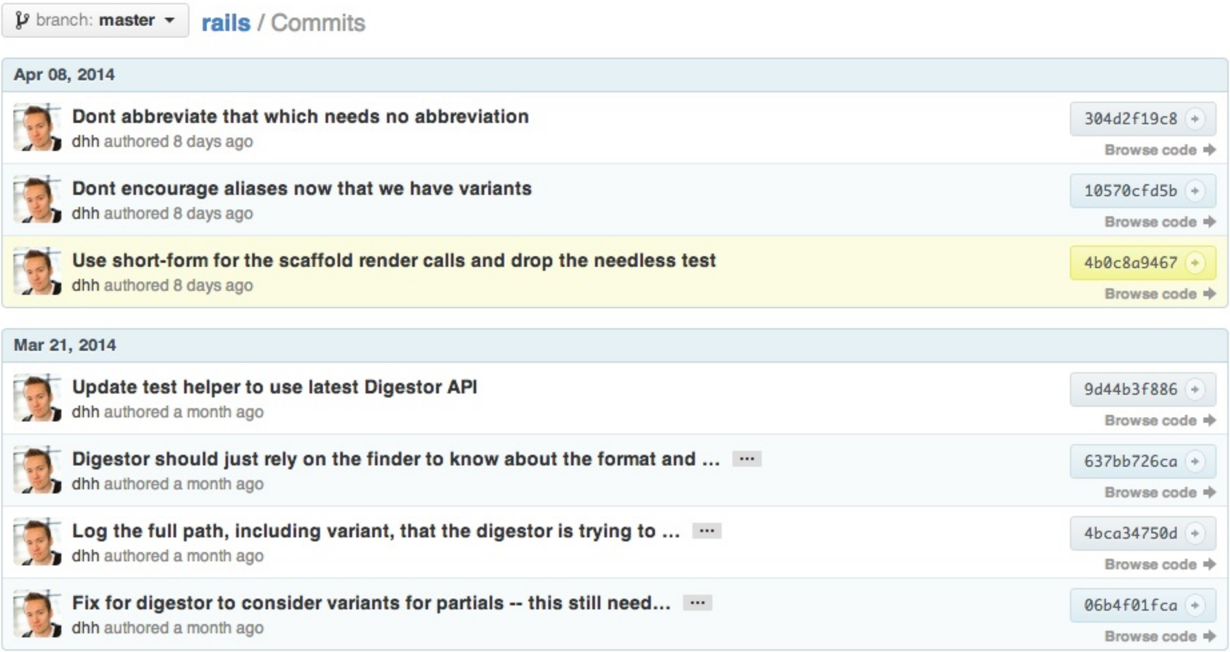
...and this is after adding `?ts=4`:



Commit History by Author

To view all commits on a repo by author add `?author=username` to the URL.

```
https://github.com/rails/rails/commits/master?author=dhh
```



Read more about the differences between commits views.

Cloning a Repository

When cloning a repository the `.git` can be left off the end.

```
$ git clone https://github.com/tiimgreen/github-cheat-sheet
```

Read more about the Git `clone` command.

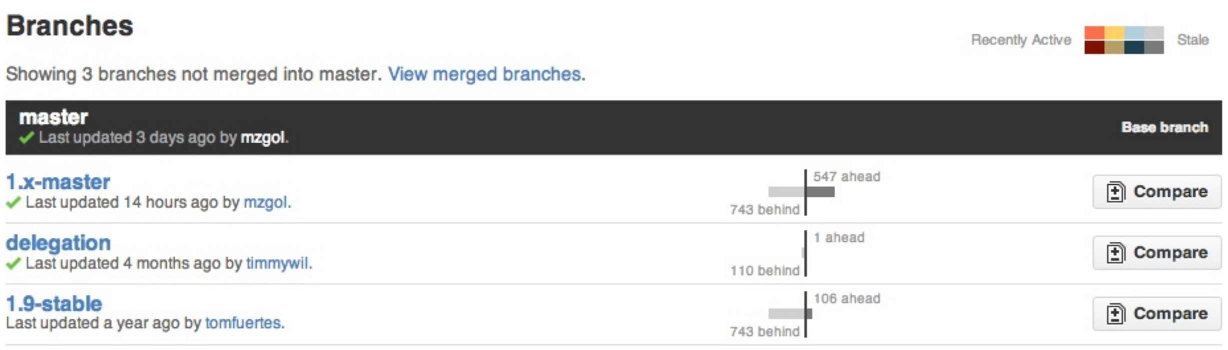
Compare all Branches to Another Branch

If you go to the repo's Branches page, next to the Commits button:

```
https://github.com/{user}/{repo}/branches
```

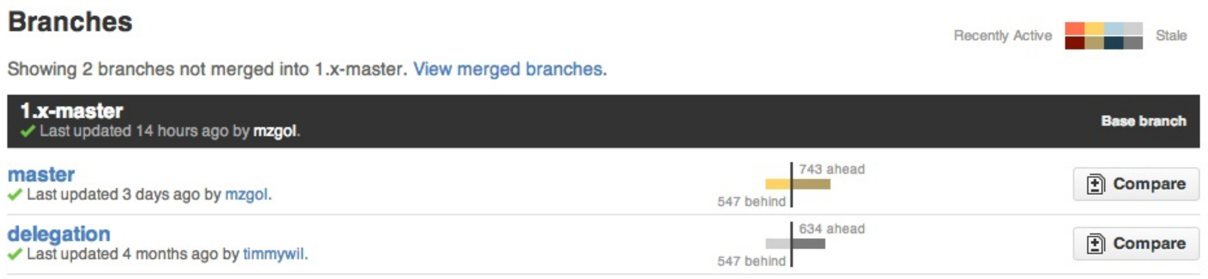
... you would see a list of all branches which are not merged into the main branch.

From here you can access the compare page or delete a branch with a click of a button.

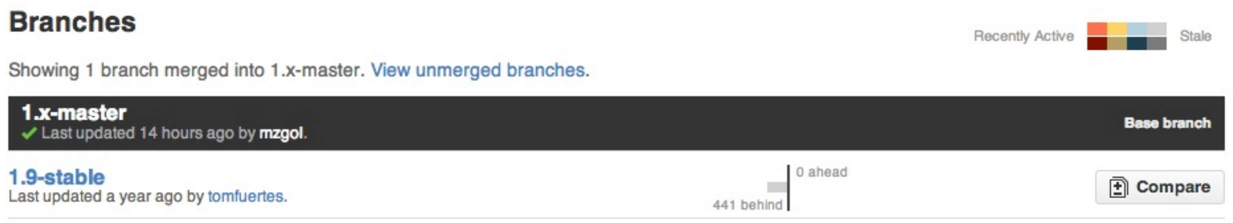


However, often you need to compare branches to a branch other than `master` (e.g. `development`). To do this, append the URL with the name of the branch like so:

```
https://github.com/{user}/{repo}/branches/{branch}
```



To see the merged branches, append `?merged=1` to the URL.



This view allows you to delete branches easily from the page, without using the command-line.

Comparing Branches

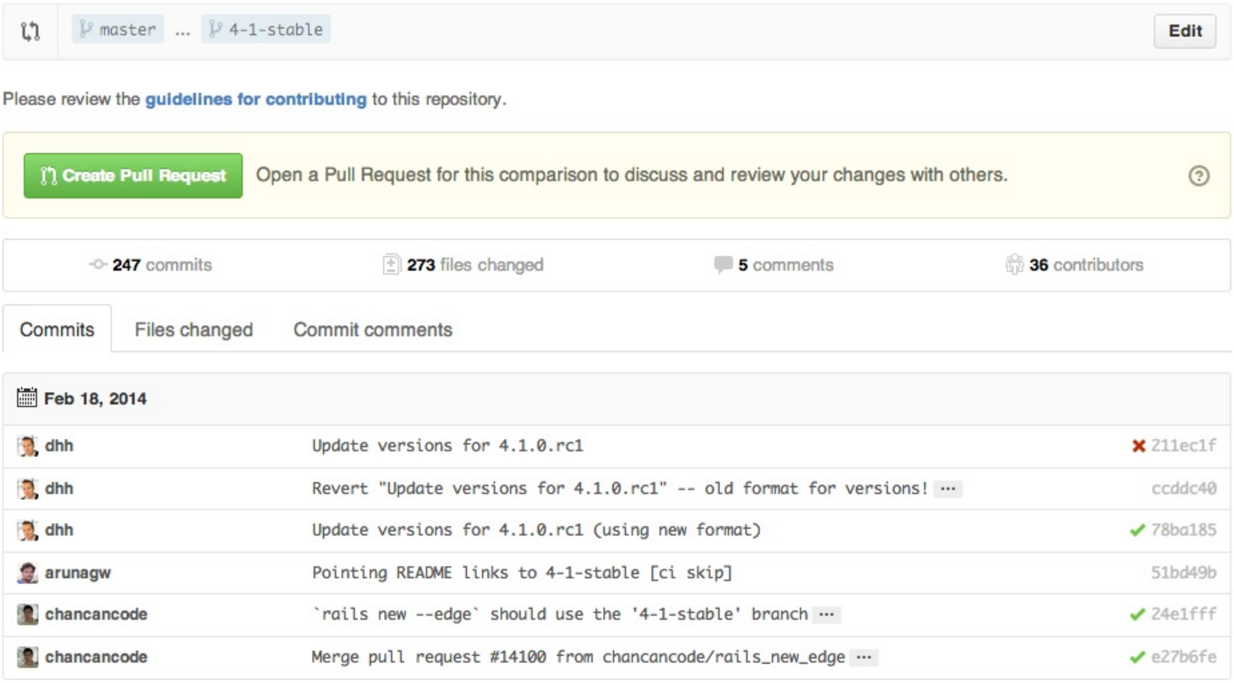
To use GitHub to compare branches, change the URL to look like this:

```
https://github.com/user/repo/compare/{range}
```

Where `{range} = master...4-1-stable`

For example:

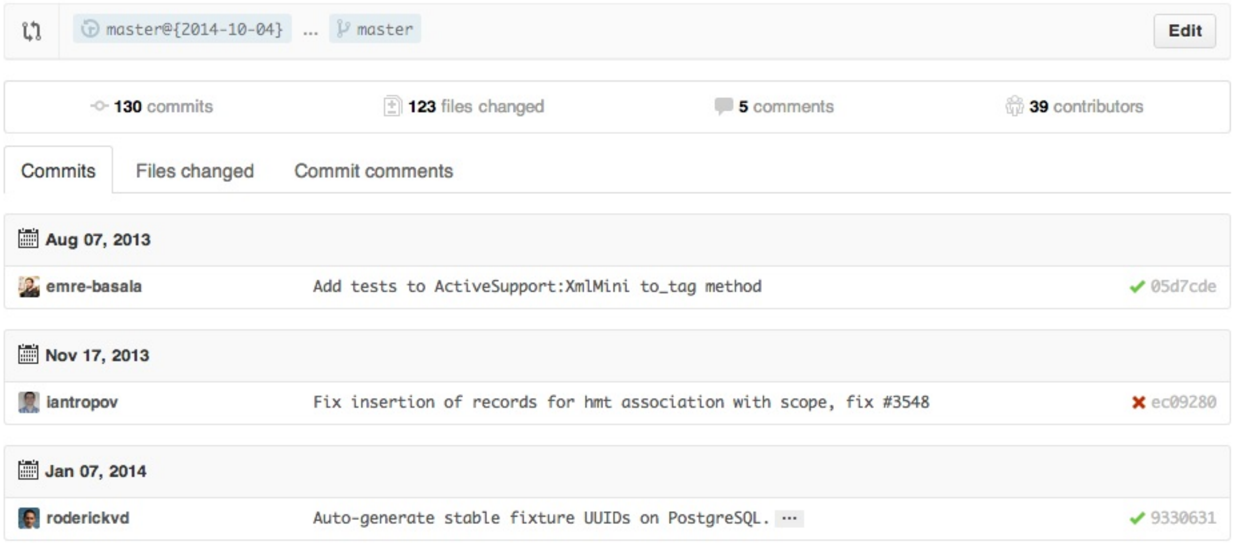
```
https://github.com/rails/rails/compare/master...4-1-stable
```



`{range}` can be changed to things like:

```
https://github.com/rails/rails/compare/master@{1.day.ago}...master
https://github.com/rails/rails/compare/master@{2014-10-04}...master
```

Dates are in the format `YYYY-DD-MM`



...which allows you to see the difference on the master branch up a set time ago or a specified date.

Read more about comparing commits across time.

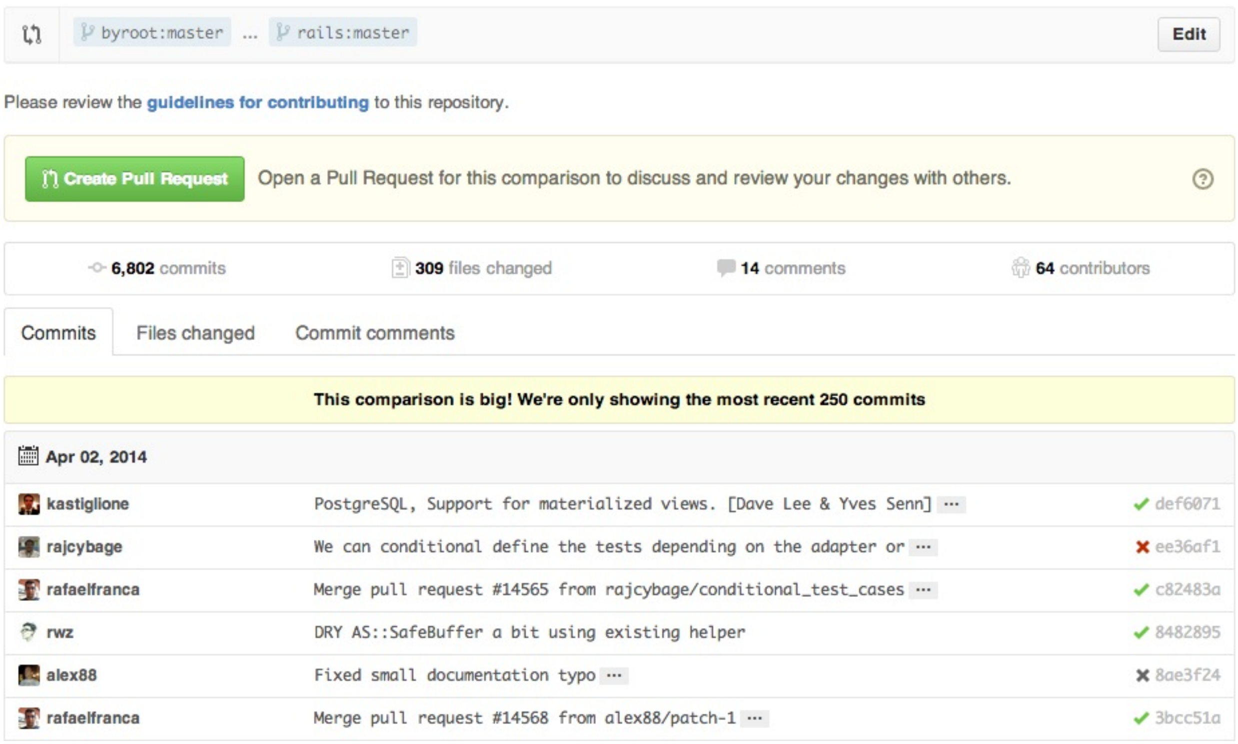
Compare Branches across Forked Repositories

To use GitHub to compare branches across forked repositories, change the URL to look like this:

```
https://github.com/user/repo/compare/{foreign-user}:{branch}...{own-branch}
```

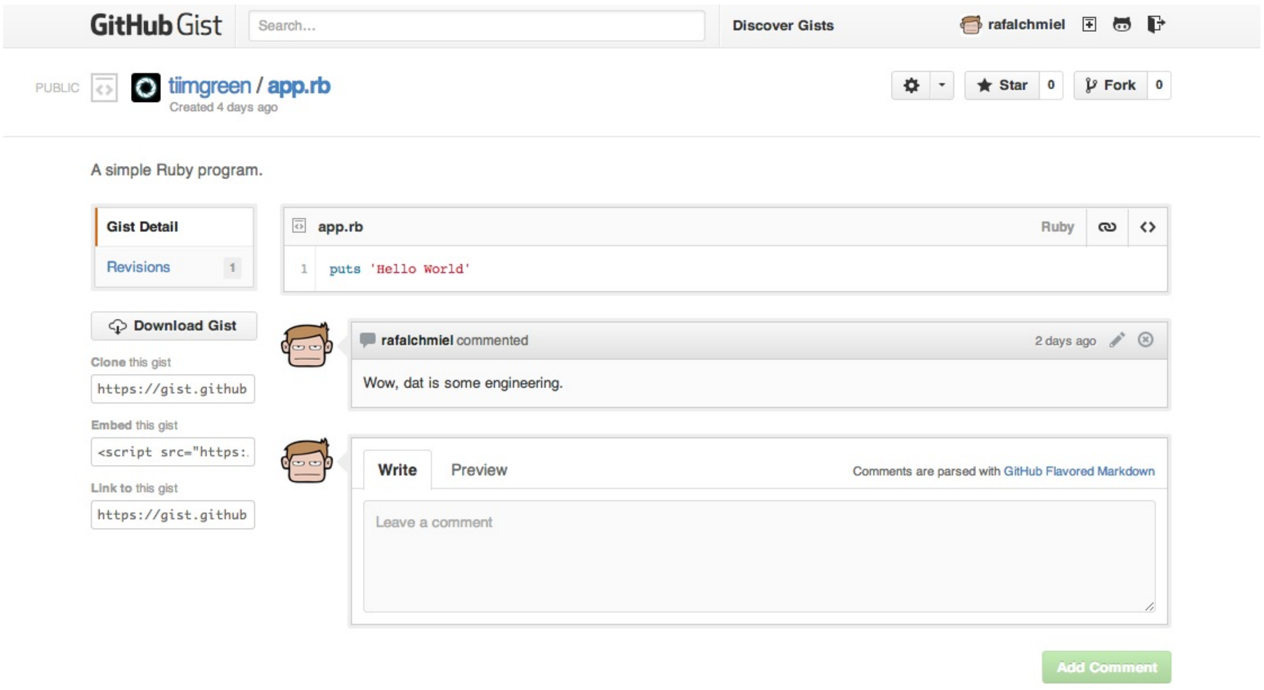
For example:

```
https://github.com/rails/rails/compare/byroot:master...master
```



Gists

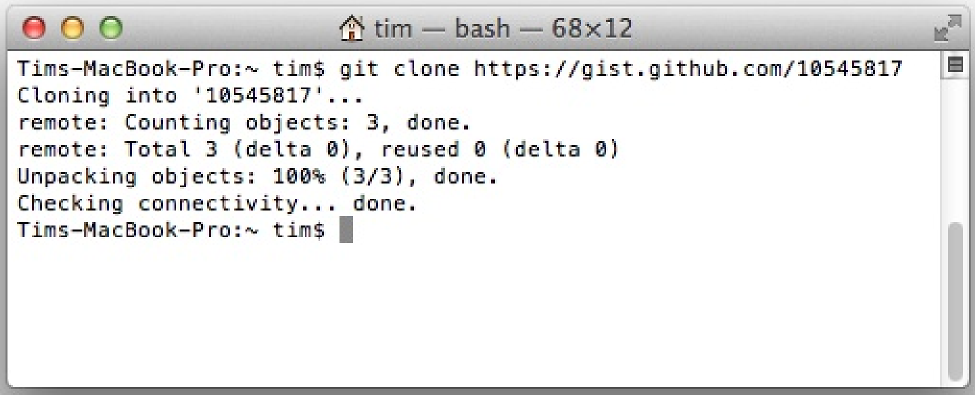
Gists are an easy way to work with small bits of code without creating a fully fledged repository.



Add `.pibb` to the end of any Gist URL (like this) in order to get the *HTML only* version suitable for embedding in any other site.

Gists can be treated as a full repository so they can be cloned like any other:

```
$ git clone https://gist.github.com/tiimgreen/10545817
```



Read more about creating gists.

Git.io

Git.io is a simple URL shortener for GitHub.

You can also use it via pure HTTP using Curl:

```
$ curl -i http://git.io -F "url=https://github.com/..."
HTTP/1.1 201 Created
Location: http://git.io/abc123

$ curl -i http://git.io/abc123
HTTP/1.1 302 Found
Location: https://github.com/...
```

Read more about Git.io.

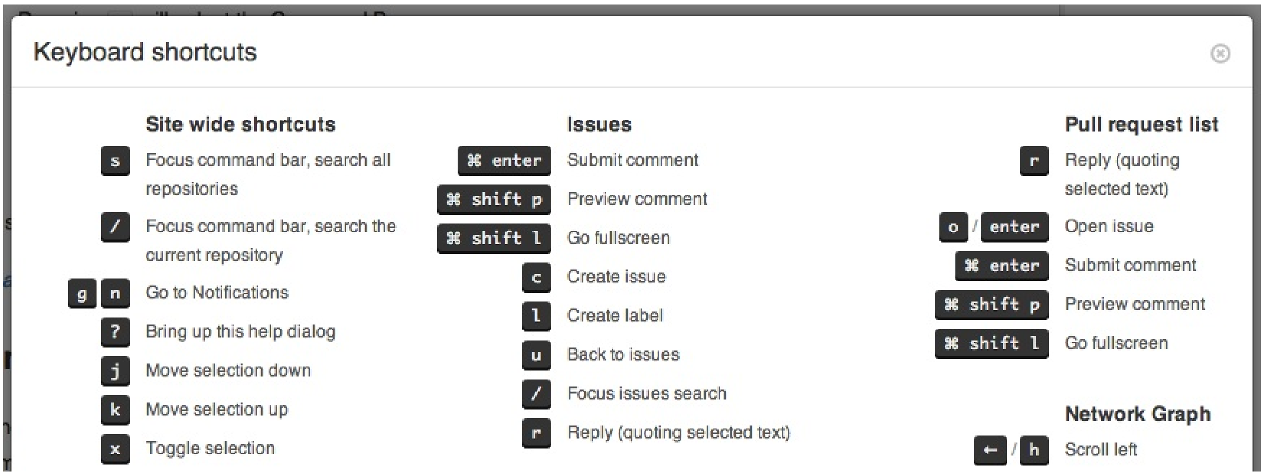
Keyboard Shortcuts

When on a repository page, keyboard shortcuts allow you to navigate easily.

- Pressing `t` will bring up a file explorer.

- Pressing `w` will bring up the branch selector.
- Pressing `s` will select the Command Bar.
- Pressing `l` will edit labels on existing Issues.
- Pressing `y` **when looking at a file** (e.g. `https://github.com/tiimgreen/github-cheat-sheet/blob/master/README.md`) will change your URL to one which, in effect, freezes the page you are looking at. If this code changes, you will still be able to see what you saw at that current time.

To see all of the shortcuts for the current page press `?`:



[Read more about using the Command Bar.]

Line Highlighting in Repositories

Either adding `#L52` to the end of a code file URL or simply clicking the line number will highlight that line number.

It also works with ranges, e.g. `#L53-L60`, to select ranges, hold `shift` and click two lines:

```
https://github.com/rails/rails/blob/master/activemodel/lib/active_model.rb#L53-L60
```

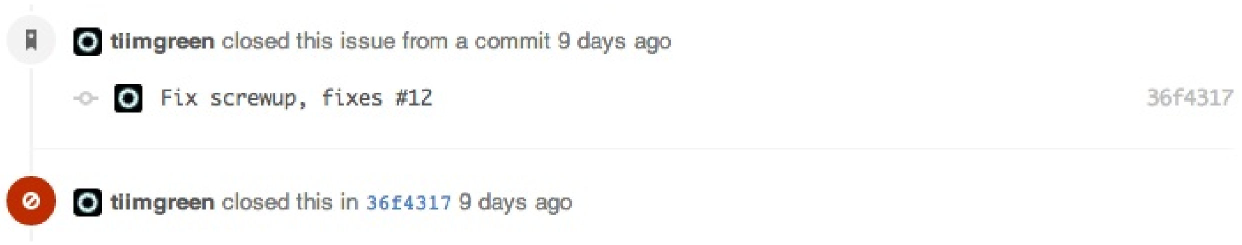
Closing Issues via Commit Messages

If a particular commit fixes an issue, any of the keywords `fix/fixes/fixed`, `close/closes/closed` or `resolve/resolves/resolved`,

followed by the issue number, will close the issue once it is committed to the master branch.

```
$ git commit -m "Fix screwup, fixes #12"
```

This closes the issue and references the closing commit.

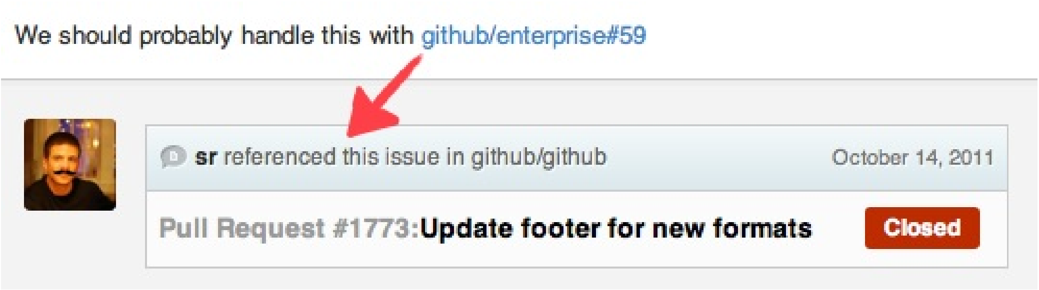


Read more about closing Issues via commit messages.

Cross-Link Issues

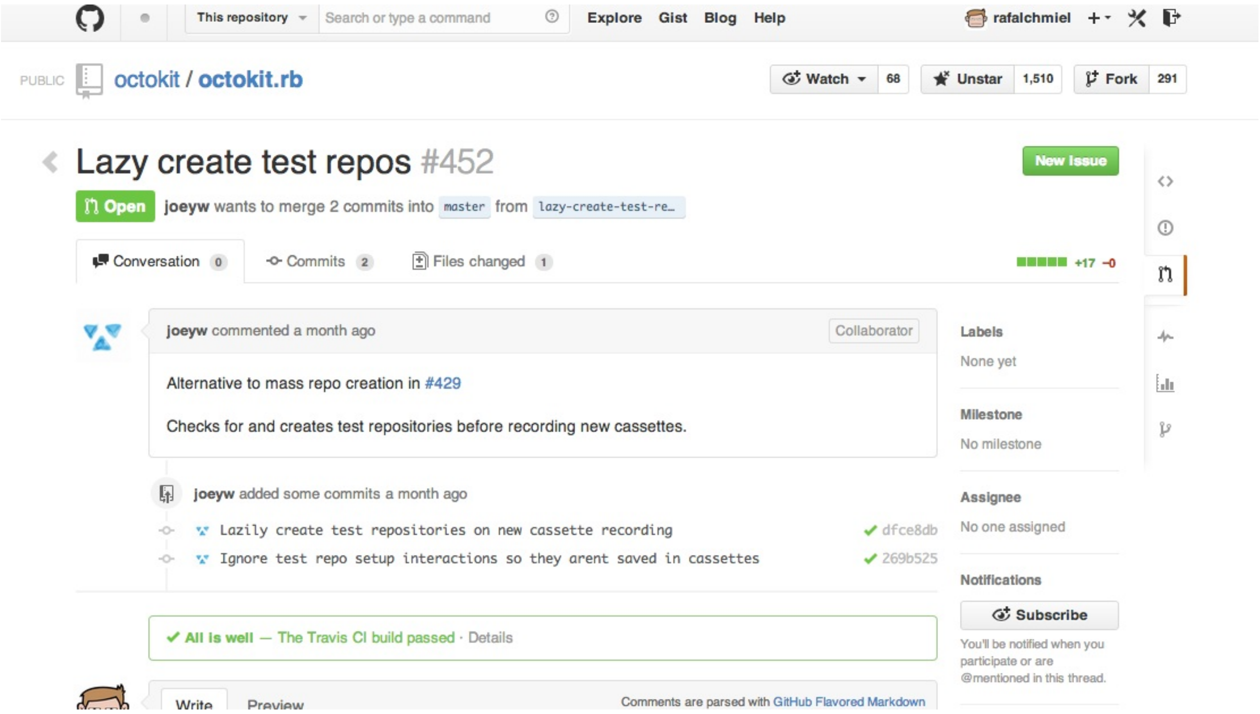
If you want to link to another issue in the same repository, simple type hash `#` then the issue number, it will be auto-linked.

To link to an issue in another repository, `user_name/repo_name#ISSUE_NUMBER` e.g. `tiimgreen/toc#12`.



CI Status on Pull Requests

If set up correctly, every time you receive a Pull Request, Travis CI will build that Pull Request just like it would every time you make a new commit. Read more about how to get started with Travis CI.



Read more about the commit status API.

Syntax Highlighting in Markdown Files

For example, to syntax highlight Ruby code in your Markdown files write:

````
```ruby
require 'tabbit'
table = Tabbit.new('Name', 'Email')
table.add_row('Tim Green', 'tiimgreen@gmail.com')
puts table.to_s
```
````

This will produce:

```ruby
require 'tabbit'
table = Tabbit.new('Name', 'Email')
table.add_row('Tim Green', 'tiimgreen@gmail.com')
puts table.to_s
```

GitHub uses Linguist to perform language detection and syntax highlighting. You can find out which keywords are valid by perusing the languages YAML file.

Read more about GitHub Flavored Markdown.

Emojis

Emojis can added to on Pull Requests, Issues, commit messages, Markdown files, etc. using `:name_of_emoji:`:

```
:smile:
```

Would produce:

:smile:

The full list of supported Emojis on GitHub can be found at emoji-cheat-sheet.com or scotch-io/All-Github-Emoji-Icons.

The top 5 used Ejmojis on GitHub are:

1. :shipit: - `:shipit:`
2. :sparkles: - `:sparkles:`
3. :-1: - `:-1:`
4. :+1: - `:+1:`
5. :clap: - `:clap:`

Images/GIFs

Images and GIFs can be added to comments, READMEs etc.:

```
![Alt Text](http://www.sheawong.com/wp-content/uploads/2013/08/keephatin.gif)
```

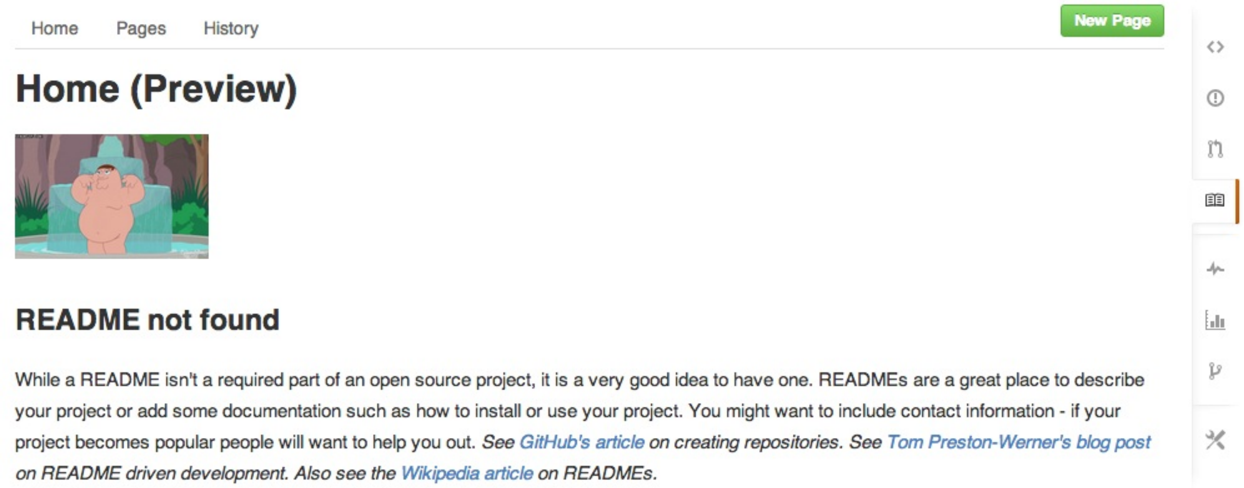
All images are cached on GitHub, so if your host goes down, the image will remain available.

Embedding Images in GitHub Wiki

There are multiple ways of embedding images in Wiki pages. There's the standard Markdown syntax (shown above). But there's also a syntax that allows things like specifying the height or width of the image:

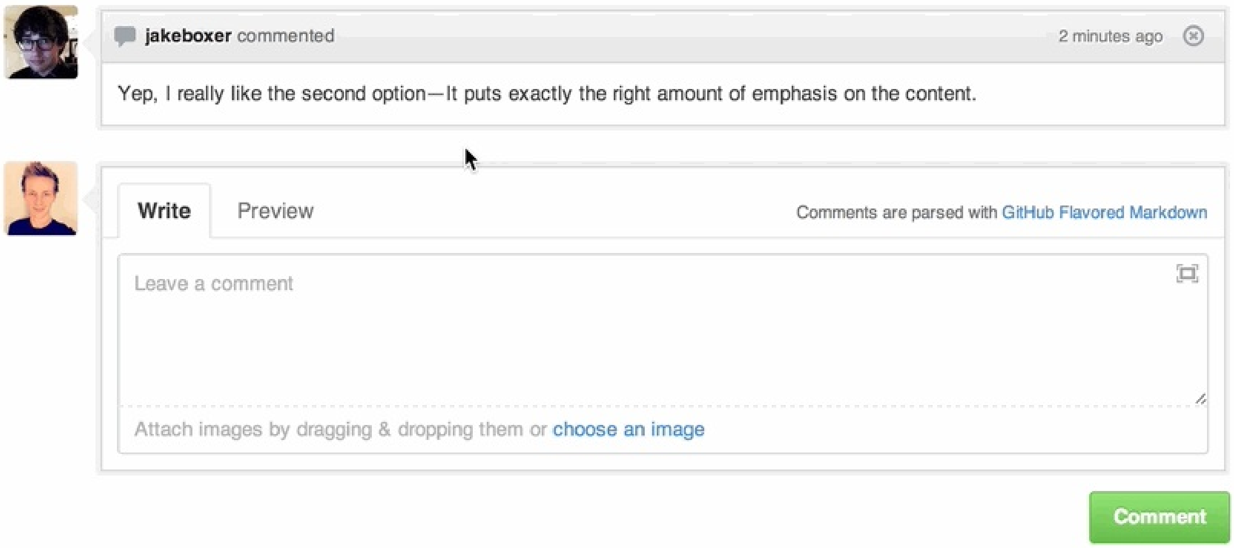
```
[[ http://www.sheawong.com/wp-content/uploads/2013/08/keephatin.gif | height = 100px ]]
```

Which produces:



Quick Quoting

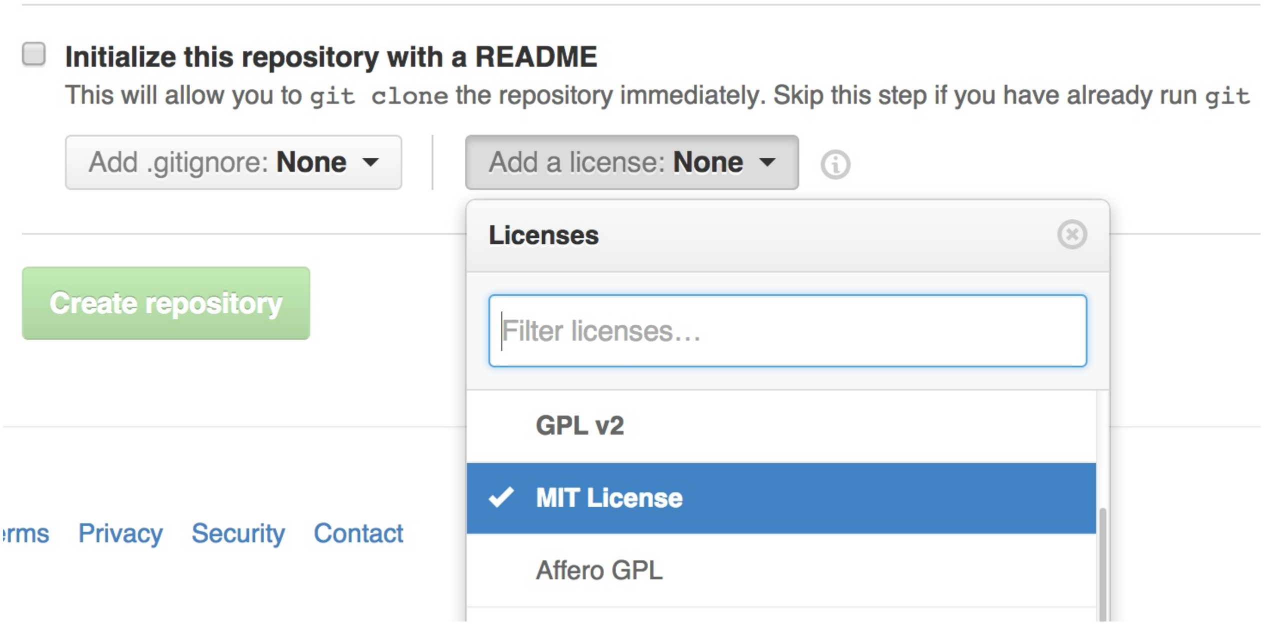
When on a comment thread and you want to quote something someone previously said, highlight the text and press `r`, this will copy it into your text box in the block-quote format.



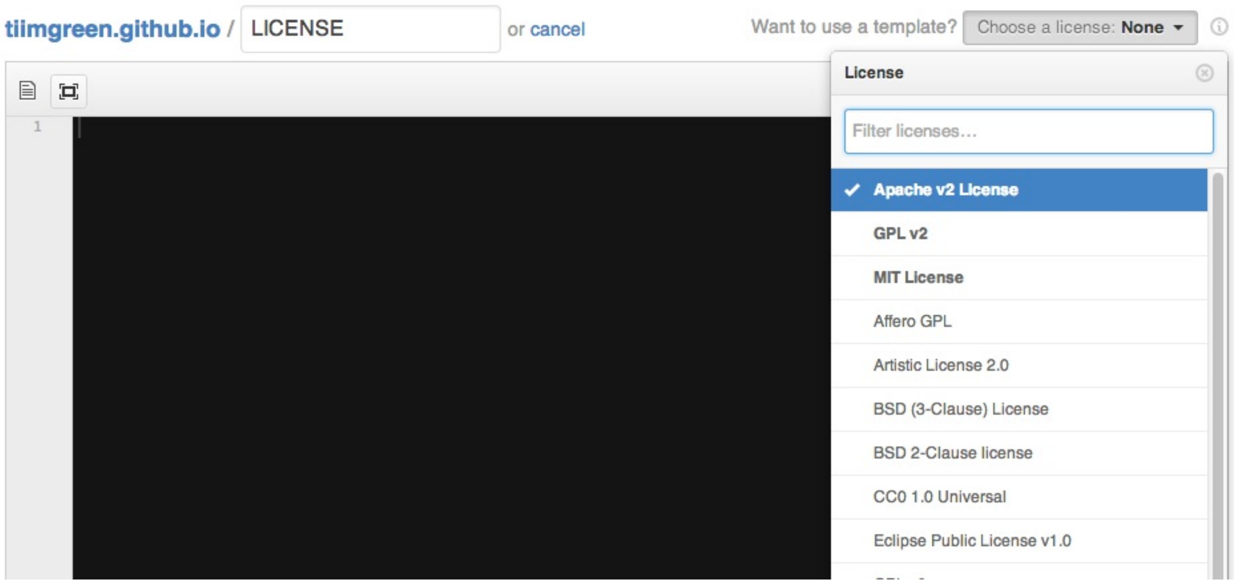
Read more about quick quoting.

Quick Licensing

When creating a repository GitHub gives you the options of adding in a pre-made license:



You can also add them to existing repositories by creating a new file through the web interface. When the name `LICENSE` is typed in you will get an option to use a template:



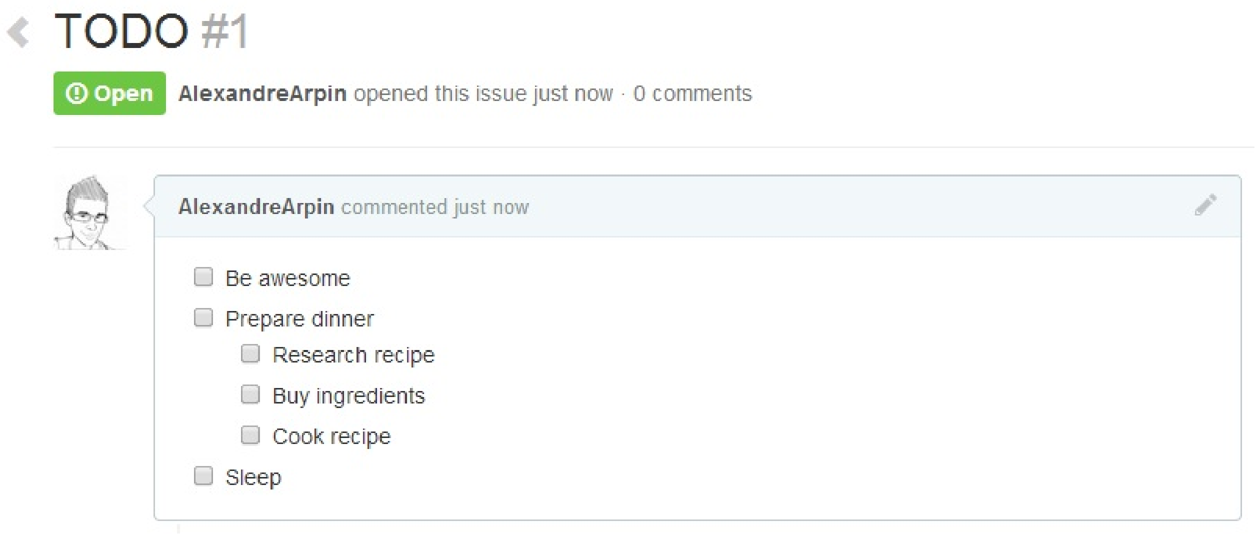
Also works for `.gitignore`.

Read more about open source licensing.

Task Lists

In Issues and Pull requests check boxes can be added with the following syntax (notice the space):

```
- [ ] Be awesome
- [ ] Prepare dinner
  - [ ] Research recipe
  - [ ] Buy ingredients
  - [ ] Cook recipe
- [ ] Sleep
```



When they are clicked, they will be updated in the pure Markdown:

```
- [x] Be awesome
- [ ] Prepare dinner
  - [x] Research recipe
  - [x] Buy ingredients
  - [ ] Cook recipe
- [ ] Sleep
```

Read more about task lists.

Task Lists in Markdown Documents

In full Markdown documents **read-only** checklists can now be added using the following syntax:

```
- [ ] Mercury
- [x] Venus
- [x] Earth
  - [x] Moon
- [x] Mars
  - [ ] Deimos
  - [ ] Phobos
```

- [ ] Mercury
- [x] Venus
- [x] Earth
  - [x] Moon
- [x] Mars
  - [ ] Deimos
  - [ ] Phobos

Read more about task lists in markdown documents.

Relative Links

Relative links are recommended in your Markdown files when linking to internal content.

```
[Link to a header](#awesome-section)
[Link to a file](docs/readme)
```

Absolute links have to be updated whenever the URL changes (e.g. repository renamed, username changed, project forked). Using relative links makes your documentation easily stand on its own.

Read more about relative links.

Metadata and Plugin Support for GitHub Pages

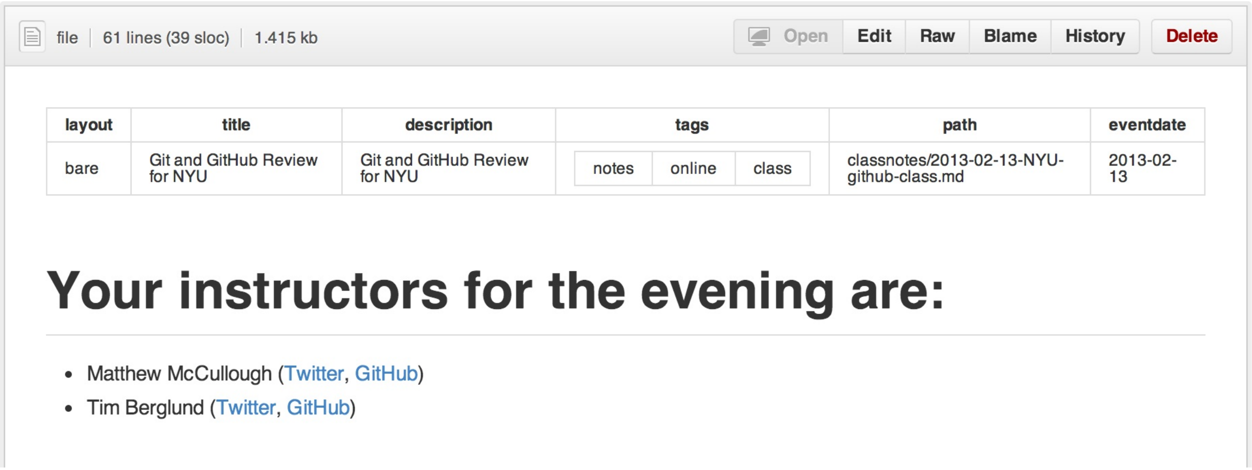
Within Jekyll pages and posts, repository information is available within the `site.github` namespace, and can be displayed, for example, using `{{ site.github.project_title }}`.

The Jemoji and jekyll-mentions plugins enable emoji and @mentions in your Jekyll posts and pages to work just like you'd expect when interacting with a repository on GitHub.com.

Read more about repository metadata and plugin support for GitHub Pages.

Viewing YAML Metadata in your Documents

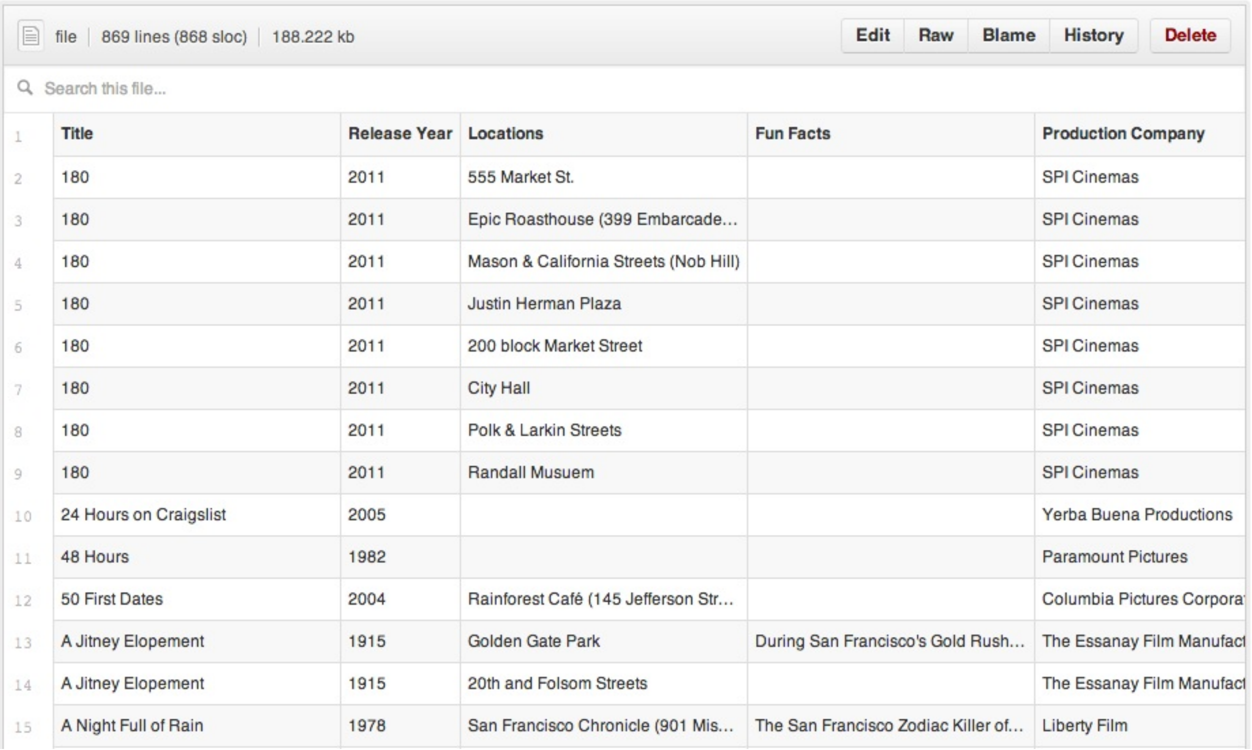
Many blogging websites, like Jekyll with GitHub Pages, depend on some YAML-formatted metadata at the beginning of your post. GitHub will render this metadata as a horizontal table, for easier reading



Read more about viewing YAML metadata in your documents.

Rendering Tabular Data

GitHub supports rendering tabular data in the form of `.csv` (comma-separated) and `.tsv` (tab-separated) files.



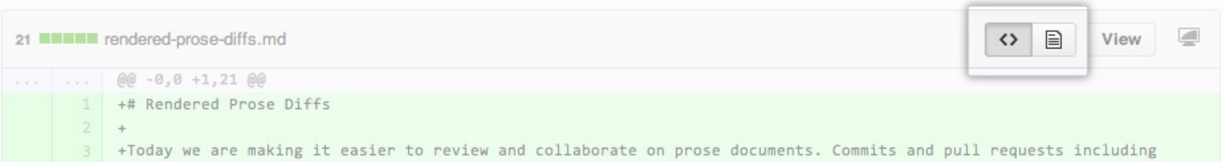
	Title	Release Year	Locations	Fun Facts	Production Company
1	Title	Release Year	Locations	Fun Facts	Production Company
2	180	2011	555 Market St.		SPI Cinemas
3	180	2011	Epic Roasthouse (399 Embarcade…		SPI Cinemas
4	180	2011	Mason & California Streets (Nob Hill)		SPI Cinemas
5	180	2011	Justin Herman Plaza		SPI Cinemas
6	180	2011	200 block Market Street		SPI Cinemas
7	180	2011	City Hall		SPI Cinemas
8	180	2011	Polk & Larkin Streets		SPI Cinemas
9	180	2011	Randall Musuem		SPI Cinemas
10	24 Hours on Craigslist	2005			Yerba Buena Productions
11	48 Hours	1982			Paramount Pictures
12	50 First Dates	2004	Rainforest Café (145 Jefferson Str…		Columbia Pictures Corpora
13	A Jitney Elopement	1915	Golden Gate Park	During San Francisco's Gold Rush…	The Essanay Film Manufact
14	A Jitney Elopement	1915	20th and Folsom Streets		The Essanay Film Manufact
15	A Night Full of Rain	1978	San Francisco Chronicle (901 Mis…	The San Francisco Zodiac Killer of…	Liberty Film

Read more about rendering tabular data.

Diffs

Rendered Prose Diffs

Commits and pull requests including rendered documents supported by GitHub (e.g. Markdown) feature *source* and *rendered* views.



Click the "rendered" button to see the changes as they'll appear in the rendered document. Rendered prose view is

handy when you're adding, removing, and editing text:

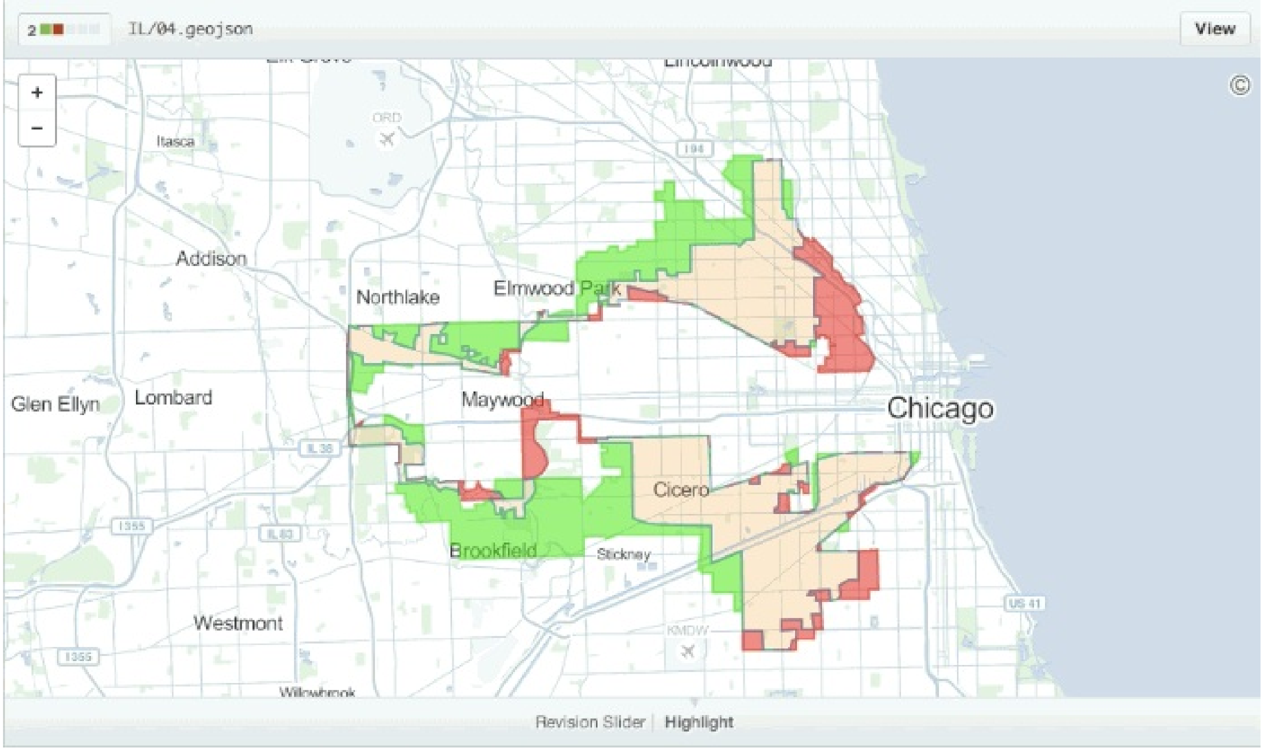
~~Obviously, you'll want to change `<your_client_id>` to match your actual Client ID.~~

Also, notice that the URL uses the `scope` query parameter to define the scopes requested by the application. For our application, we're requesting `user:email` scope for reading private email addresses.

Read more about rendered prose diffs.

Diffable Maps

Any time you view a commit or pull request on GitHub that includes geodata, GitHub will render a visual representation of what was changed.



Read more about diffable maps.

Expanding Context in Diffs

Using the *unfold* button in the gutter of a diff, you can reveal additional lines of context with a click. You can keep clicking *unfold* until you've revealed the whole file, and the feature is available anywhere GitHub renders diffs.

```
+- (RACSignal *)enqueueRequest:(NSURLRequest *)request fetchAllPages:(BOOL)fetchAllPages;
+
 // Enqueues a request to fetch information about the current user by accessing
 // a path relative to the user object.
 //
@@ -241,11 +255,13 @@ - (id)initWithServer:(OCTServer *)server {
   NSString *userAgent = self.class.userAgent;
   if (userAgent != nil) [self setDefaultHeader:@"User-Agent" value:userAgent];

-  self.parameterEncoding = AFJSONParameterEncoding;
-  [self setDefaultHeader:@"Accept" value:@"application/vnd.github.beta+json"];
-
   [AFHTTPRequestOperation addAcceptableStatusCodes:[NSIndexSet indexSetWithIndex:OCTClientNotModifiedStatusCode]]
-  [AFJSONRequestOperation addAcceptableContentTypes:[NSSet setWithObject:@"application/vnd.github.beta+json"]];
+
+  NSString *contentType = [NSString stringWithFormat:@"application/vnd.github.%@+json", OCTClientAPIVersion];
+  [self setDefaultHeader:@"Accept" value:contentType];
+  [AFJSONRequestOperation addAcceptableContentTypes:[NSSet setWithObject:contentType]];
+
+  self.parameterEncoding = AFJSONParameterEncoding;
   [self registerHTTPOperationClass:AFJSONRequestOperation.class];

   return self;
```

Read more about expanding context in diffs.

Diff or Patch of Pull Request

You can get the diff of a Pull Request by adding a `.diff` or `.patch` extension to the end of the URL. For example:

```
https://github.com/tiimgreen/github-cheat-sheet/pull/15
https://github.com/tiimgreen/github-cheat-sheet/pull/15.diff
https://github.com/tiimgreen/github-cheat-sheet/pull/15.patch
```

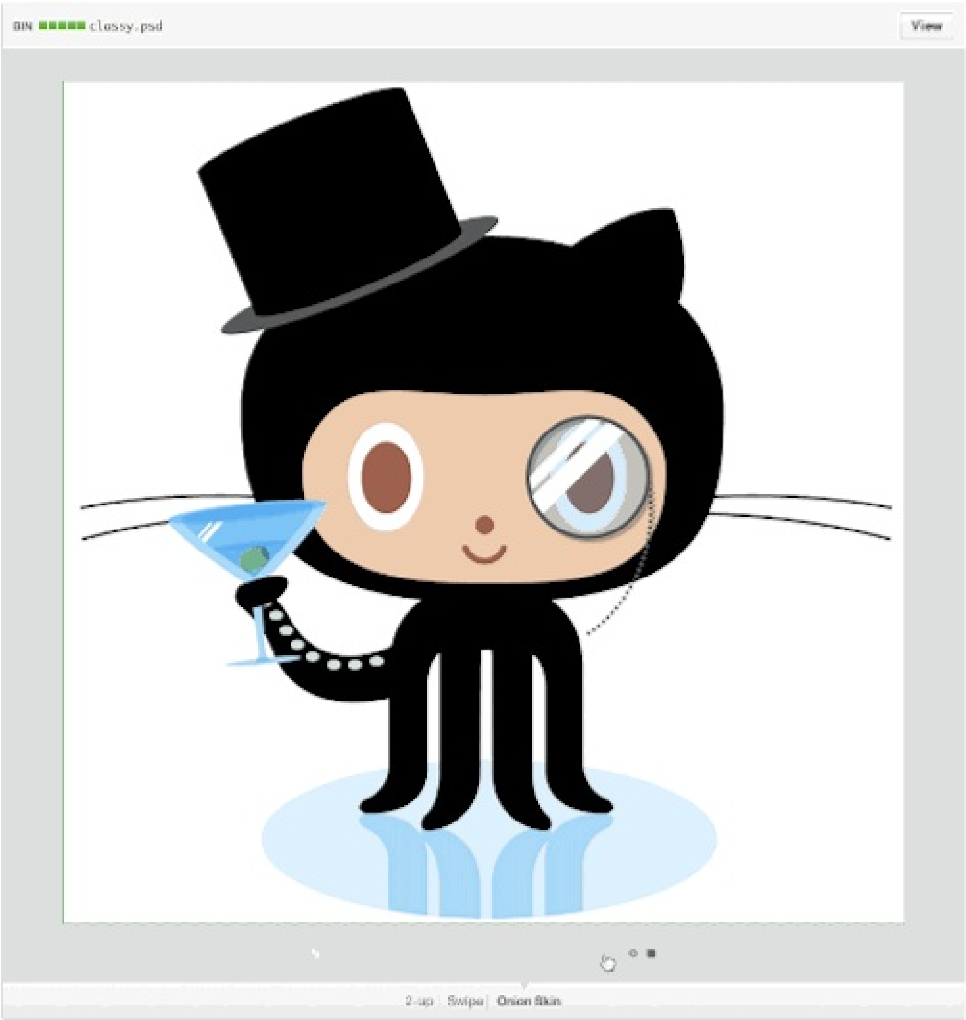
The `.diff` extension would give you this in plain text:

```
diff --git a/README.md b/README.md
index 88fcf69..8614873 100644
--- a/README.md
+++ b/README.md
@@ -28,6 +28,7 @@ All the hidden and not hidden features of Git and GitHub. This cheat sheet was i
 - [Merged Branches](#merged-branches)
 - [Quick Licensing](#quick-licensing)
 - [TODO Lists](#todo-lists)
+- [Relative Links](#relative-links)
 - [.gitconfig Recommendations](#gitconfig-recommendations)
     - [Aliases](#aliases)
     - [Auto-correct](#auto-correct)
@@ -381,6 +382,19 @@ When they are clicked, they will be updated in the pure Markdown:
 - [ ] Sleep

(...)
```

Rendering and diffing images

GitHub can display several common image formats, including PNG, JPG, GIF, and PSD. In addition, there are several ways to compare differences between versions of those image formats.



Read more about rendering and diffing images.

Hub

Hub is a command line Git wrapper that gives you extra features and commands that make working with GitHub easier.

This allows you to do things like:

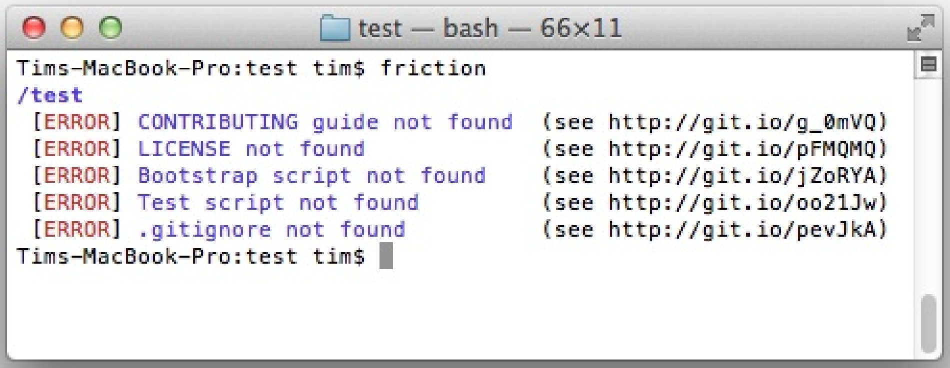
```
$ hub clone tiimgreen/toc
```

Check out some more cool commands Hub has to offer.

Decreasing Contributor Friction

If you want people to use and contribute to your project, you need to start by answering their most basic questions. What does the project do? How do I use it? How am I allowed to use it? How do I contribute? How do I get up and running in development? How do I make sure my new features didn't break old functionality?

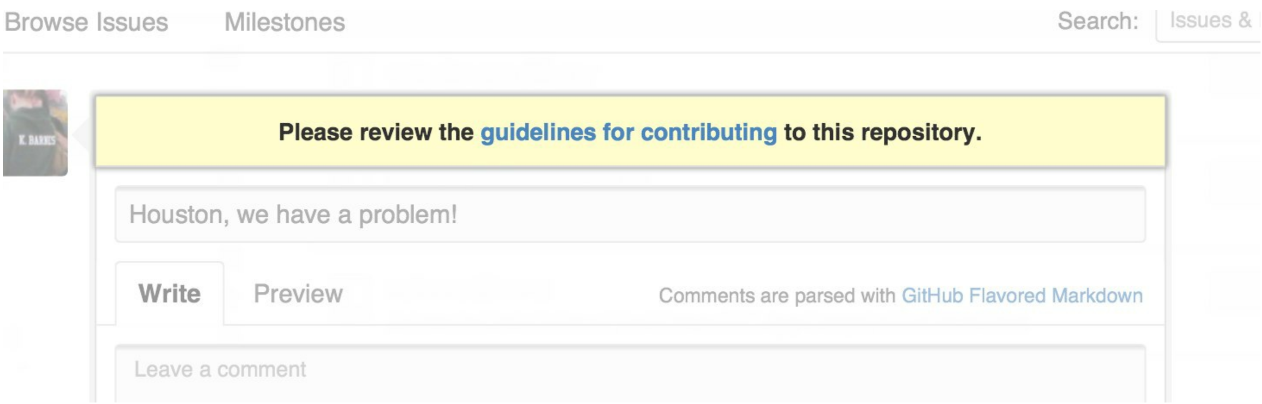
Friction is a command line script that will check your project for common answers to these questions. This is some example output:



Friction supports MRI 2.1.0, MRI 2.0.0, and MRI 1.9.3.

Contributing Guidelines

Adding a `CONTRIBUTING` file to the root of your repository will add a link to your file when a contributor creates an Issue or opens a Pull Request.



Read more about contributing guidelines.

GitHub Resources

Title	Link
GitHub Explore	https://github.com/explore
GitHub Blog	https://github.com/blog
GitHub Help	https://help.github.com/
GitHub Training	http://training.github.com/
GitHub Developer	https://developer.github.com/

GitHub Talks

Title	Link
How GitHub Uses GitHub to Build GitHub	https://www.youtube.com/watch?v=qyz3jkOBbQY
Introduction to Git with Scott Chacon of GitHub	https://www.youtube.com/watch?v=ZDR433b0HJY

How GitHub No Longer Works Git and GitHub Secrets	https://www.youtube.com/watch?v=gXD1ITW7iZI https://www.youtube.com/watch?v=Foz9yvMkvlA
More Git and GitHub Secrets	https://www.youtube.com/watch?v=p50xsL-iVgU

Git

Previous Branch

To move to the previous branch in Git:

```
$ git checkout -
# Switched to branch 'master'

$ git checkout -
# Switched to branch 'next'

$ git checkout -
# Switched to branch 'master'
```

Read more about Git branching.

Stripspace

Git Stripspace:

- Strips trailing whitespace
- Collapses newlines
- Adds newline to end of file

A file must be passed when calling the command, e.g.:

```
$ git stripspace < README.md
```

Read more about the Git `stripspace` command.

Checking out Pull Requests

Pull Requests are special branches on the GitHub repository which can be retrieved locally in several ways:

Retrieve a specific Pull Request and store it temporarily in `FETCH_HEAD` for quickly `diff`ing or `merge`ing:

```
$ git fetch origin refs/pull/[PR-Number]/head
```

Acquire all Pull Request branches as local remote branches by refspec:

```
$ git fetch origin '+refs/pull/*/head:refs/remotes/origin/pr/*'
```

Or setup the remote to fetch Pull Requests automatically by adding these corresponding lines in your repository's `.git/config`:

```
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = git@github.com:tiimgreen/github-cheat-sheet.git
```

```
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = git@github.com:tiimgreen/github-cheat-sheet.git
    fetch = +refs/pull/*/head:refs/remotes/origin/pr/*
```

For Fork-based Pull Request contributions, it's useful to `checkout` a remote branch representing the Pull Request and create a local branch from it:

```
$ git checkout pr/42 pr-42
```

Read more about checking out pull requests locally.

Empty Commits :trollface:

Commits can be pushed with no code changes by adding `--allow-empty`:

```
$ git commit -m "Big-ass commit" --allow-empty
```

Some use-cases for this (that make sense), include:

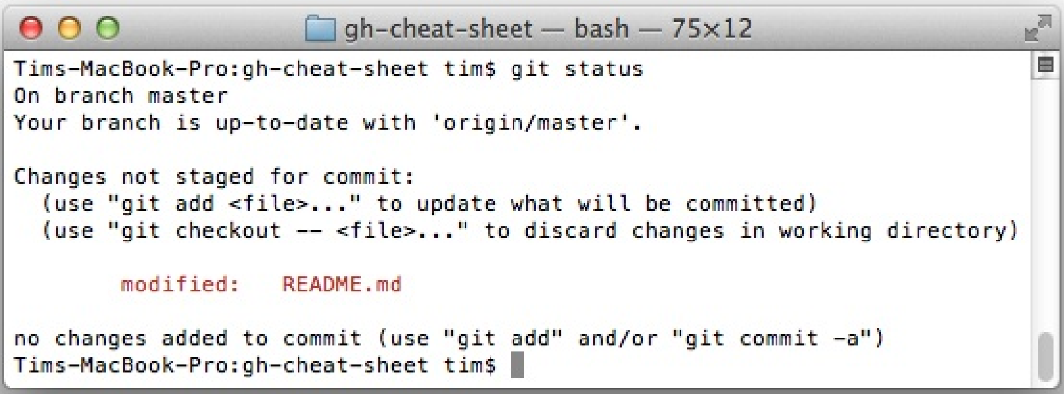
- Annotating the start of a new bulk of work or a new feature.
- Documenting when you make changes to the project that aren't code related.
- Communicating with people using your repository.
- The first commit of a repo, as the first commit cannot be rebased later: `git commit -m "init repo" --allow-empty`.

Styled Git Status

Running:

```
$ git status
```

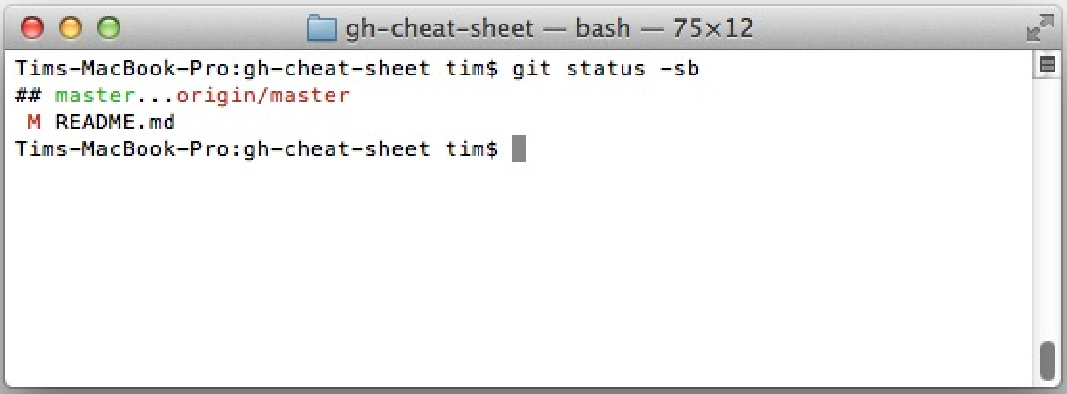
Produces:



By adding `-sb`:

```
$ git status -sb
```

This is produced:



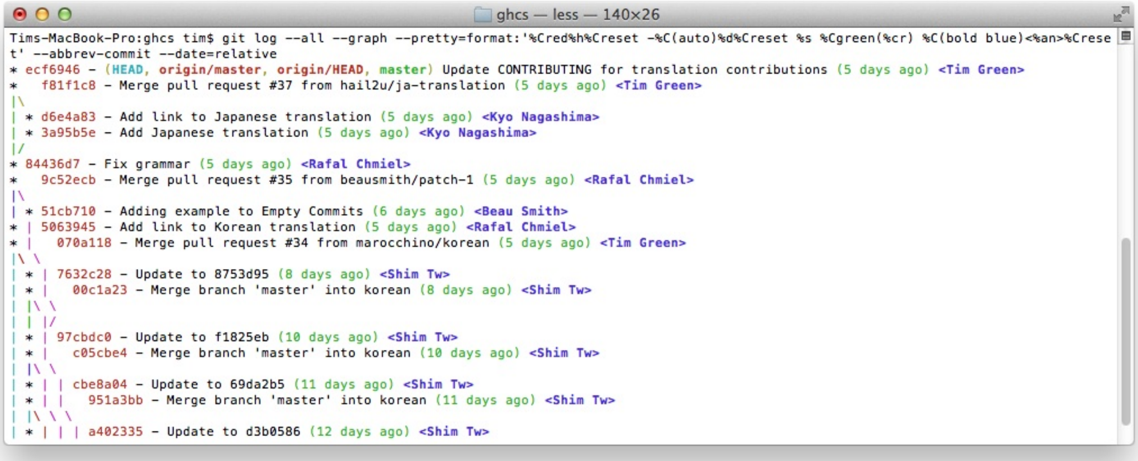
Read more about the Git `status` command.

Styled Git Log

Running:

```
$ git log --all --graph --pretty=format:'%Cred%h%Creset -%C(auto)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abb
```

Produces:



Credit to Palesz

This can be aliased using the instructions found here.

Read more about the Git `log` command.

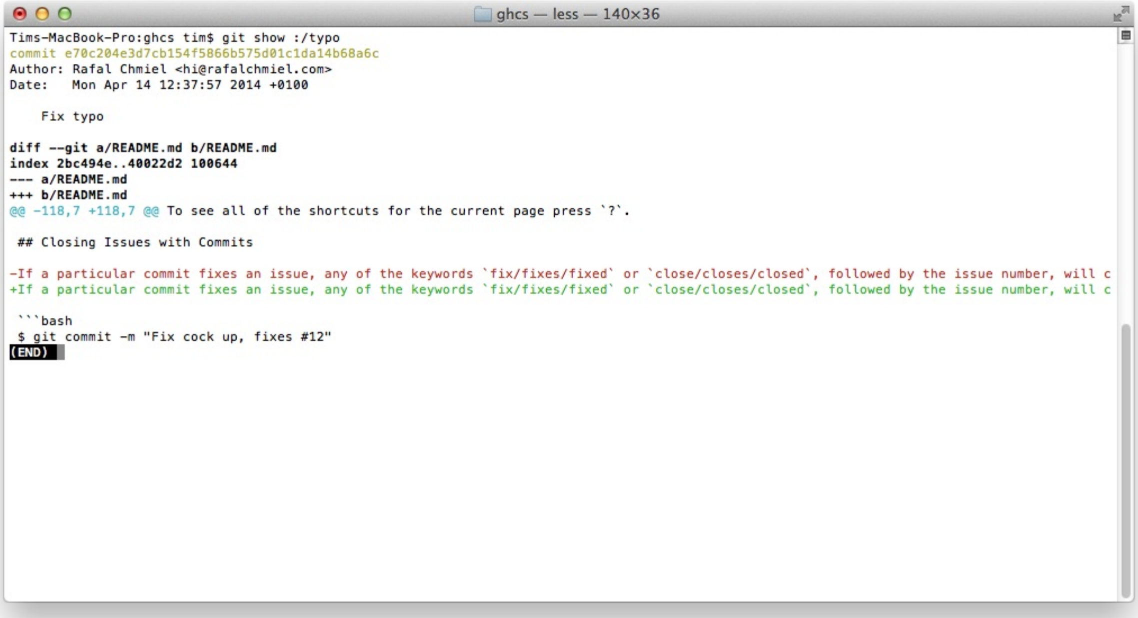
Git Query

A Git query allows you to search all your previous commit messages and find the most recent one matching the query.

```
$ git show :/query
```

Where `query` (case-sensitive) is the term you want to search, this then finds the last one and gives details on the lines that were changed.

```
$ git show :/typo
```



Press `q` *to quit.*

Merged Branches

Running:

```
$ git branch --merged
```

Will give you a list of all branches that have been merged into your current branch.

Conversely:

```
$ git branch --no-merged
```

Will give you a list of branches that have not been merged into your current branch.

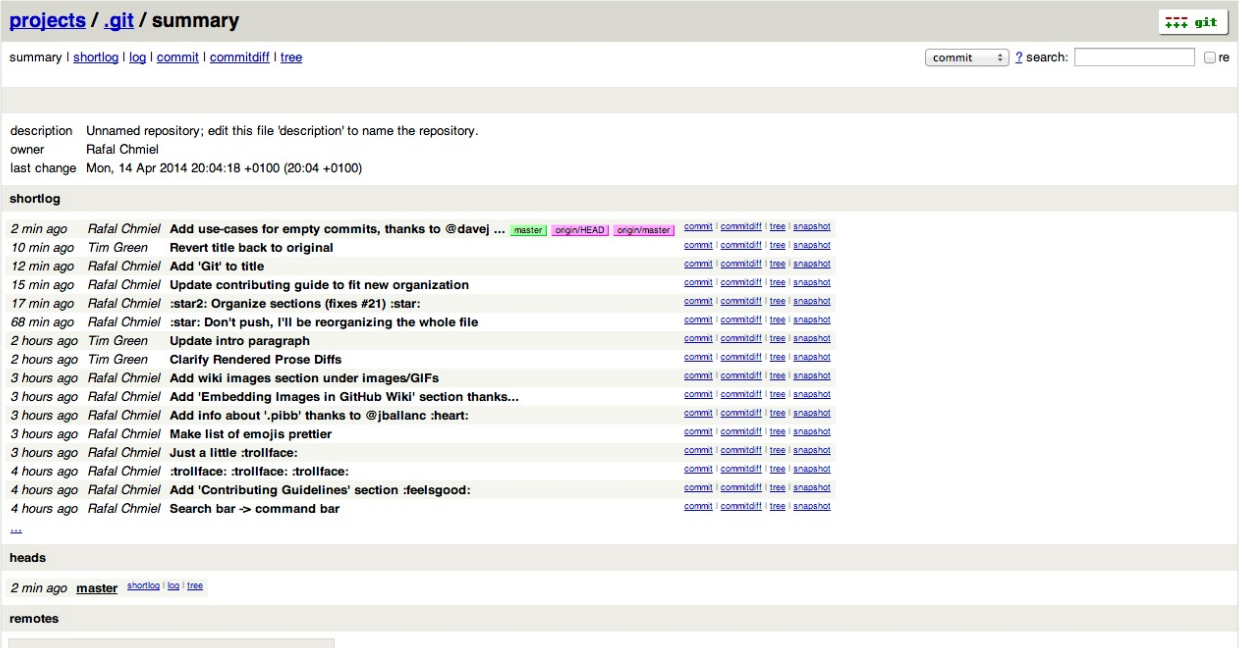
Read more about the Git `branch` *command.*

Web Server for Browsing Local Repositories

Use the Git `instaweb` command to instantly browse your working repository in `gitweb`. This command is a simple script to set up `gitweb` and a web server for browsing the local repository.

```
$ git instaweb
```

Opens:



Read more about the Git `instaweb` command.

Git Configurations

Your `.gitconfig` file contains all your Git configurations.

Aliases

Aliases are helpers that let you define your own git calls. For example you could set `git a` to run `git add --all`.

To add an alias, either navigate to `~/.gitconfig` and fill it out in the following format:

```
[alias]
  co = checkout
  cm = commit
  p = push
  # Show verbose output about tags, branches or remotes
  tags = tag -l
  branches = branch -a
  remotes = remote -v
```

...or type in the command-line:

```
$ git config --global alias.new_alias git_function
```

For example:

```
$ git config --global alias.cm commit
```

For an alias with multiple functions use quotes:

```
$ git config --global alias.ac 'add -A . && commit'
```

Some useful aliases include:

Alias	Command	What to Type
git cm	git commit	git config --global alias.cm commit
git co	git checkout	git config --global alias.co checkout
git ac	git add . -A git commit	git config --global alias.ac '!git add -A && git commit'
git st	git status -sb	git config --global alias.st 'status -sb'
git tags	git tag -l	git config --global alias.tags 'tag -l'
git branches	git branch -a	git config --global alias.branches 'branch -a'
git remotes	git remote -v	git config --global alias.remotes 'remote -v'
git lg	git log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --	git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit --"

Some Aliases are taken from @mathiasbynens dotfiles: https://github.com/mathiasbynens/dotfiles/blob/master/.gitconfig

Auto-Correct

If you type `git comit` you will get this:

```
$ git comit -m "Message"
# git: 'comit' is not a git command. See 'git --help'.

# Did you mean this?
#   commit
```

To call `commit` when `comit` is typed, just enable auto-correct:

```
$ git config --global help.autocorrect 1
```

So now you will get this:

```
$ git comit -m "Message"
# WARNING: You called a Git command named 'comit', which does not exist.
# Continuing under the assumption that you meant 'commit'
# in 0.1 seconds automatically...
```

Color

To add more color to your Git output:

```
$ git config --global color.ui 1
```

Read more about the Git `config` command.

Git Resources

Title	Link
Official Git Site	http://git-scm.com/
Official Git Video Tutorials	http://git-scm.com/videos
Code School Try Git	http://try.github.com/
Introductory Reference & Tutorial for Git	http://gitref.org/
Official Git Tutorial	http://git-scm.com/docs/gittutorial
Everyday Git	http://git-scm.com/docs/everyday
Git Immersion	http://gitimmersion.com/
Ry's Git Tutorial	http://rypress.com/tutorials/git/index.html
Git for Designer	http://hoth.entp.com/output/git_for_designers.html
Git for Computer Scientists	http://eagain.net/articles/git-for-computer-scientists/
Git Magic	http://www-cs-students.stanford.edu/~blynn/gitmagic/
GitHub Training Kit	http://training.github.com/kit

Git Books

Title	Link
Pragmatic Version Control Using Git	http://www.pragprog.com/titles/tsgit/pragmatic-version-control-using-git
Pro Git	http://git-scm.com/book
Git Internals Peepcode	http://peepcode.com/products/git-internals-pdf
Git in the Trenches	http://cbx33.github.com/gitt/
Version Control with Git	http://www.amazon.com/Version-Control-Git-collaborative-development/dp/1449316387
Pragmatic Guide to Git	http://www.pragprog.com/titles/pg_git/pragmatic-guide-to-git
Git: Version Control for Everyone	http://www.packtpub.com/git-version-control-for-everyone/book